High Speed Low Power Viterbi Decoder Design for TCM

Maram.Saritha Reddy
Nagole institute of technology & science
Department of electronics and communication engineering

K.Srinivasa Reddy
Nagole institute of technology & science
Department of electronics and communication engineering

G.Ravikumar
Nagole institute of technology & science
Department of electronics and communication engineering

Abstract—High-speed, low-power design of Viterbi decoders for trellis coded modulation (TCM) systems is presented in this paper. It is well known that the Viterbi decoder (VD) is the dominant module determining the overall power consumption of TCM decoders. We propose a pre-computation architecture incorporated with T -algorithm for VD, which can effectively reduce the power consumption without degrading the decoding speed much.

Index Terms—Trellis coded modulation (TCM), viterbi decoder, VLSI.

I. INTRODUCTION

Trellis coded modulation (TCM) schemes are used in many bandwidth- efficient systems. Typically, a TCM system employs a high-rate convolutional code, which leads to a high complexity of the Viterbi decoder (VD) for the TCM decoder, even if the constraint length of the convolutional code is moderate. For example, the rate-3/4 convolutional code used in a 4-D TCM system for deep space communications has a constraint length of 7; however, the computational complexity of the correspondingVD is equivalent to that of a VD for a rate-1/2 convolutional code with a constraint length of 9 due to the large number of transitions in the trellis. Therefore, in terms of power consumption, the Viterbi decoder is the dominant module in a TCM decoder. In order to reduce the computational complexity as well as the power consumption, low-power schemes should be exploited for the VD in a TCM decoder.

General solutions for low-power VD design have been well studied by existing work. Power reduction in VDs could be achieved by reducing the number of states (for example, reduced-state sequence decoding (RSSD), M-algorithm and T -algorithm or by over-scaling the supply voltage. Over-scaling of the supply voltage usually needs to take into consideration the whole system that includes the VD (whether the system allows such an over-scaling or not), which is not the main focus of our research. RSSD is in general not as efficient as the M-algorithm and T-algorithm is more commonly used than M-algorithm in practical applications, because the M-algorithm requires a sorting process in a feedback loop while T-algorithm only searches for the optimal path metric (PM), that is, the minimum value or the maximum value of all PMs.

T-algorithm has been shown to be very efficient in reducing the power consumption .However, searching for the optimal PM in the feedback loop still reduces the decoding speed. To overcome this drawback, two variations of the T -algorithm have been proposed: the relaxed adaptive VD , which suggests using an estimated optimal PM, instead of finding the real one each cycle and the limited-search parallel state VD based on scarce state transition (SST) . In our preliminary work , we have shown that when applied to high-rate convolutional codes, the relaxed adaptive VD suffers a severe degradation of bit-error-rate (BER) performance due to the inherent drifting error between the estimated optimal PM and the accurate one.

II. VITERBI DECODER

A typical functional block diagram of a Viterbi decoder is shown in Fig. 1. First, branch metrics (BMs) are calculated in the BM unit (BMU) from the received symbols. In a TCM decoder, this module is replaced by transition metrics unit (TMU), which is more complex than the BMU. Then, BMs are fed into the ACSU that recursively computes the PMs and outputs decision bits for each possible state transition.

After that, the decision bits are stored in and retrieved from the SMU in order to decode the source bits along the final survivor path. The PMs of the current iteration are stored in the PM unit (PMU). T-algorithm requires extra computation in the ACSU loop for calculating the optimal PM and puncturing states. Therefore, a straightforward implementation of T -algorithm will dramatically reduce the decoding speed.

The key point of improving the clock speed of M –algorithm is to quickly find the optimal PM.

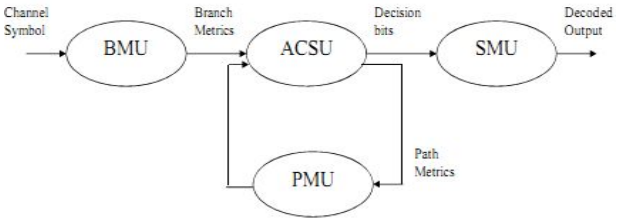


Figure 1: Viterbi decoder block diagram.

III IMPLEMENTATION OF VITERBI DECODER

The implementation of the Viterbi decoder, a processor that implements the Viterbi algorithm, consists of three major blocks: the branch metrics calculation unit (BMU), the add-compare-select unit (ACS), and the survivor path decoding unit.

A typical viterbi decoder block diagram is shown in Figure, First branch metrics (BMs) are calculated in the BM unit (BMU) from the received symbols. In a TCM decoder, this module is replaced by transition metric unit (TMU), which is more complex than BMU. Then BMs are fed into the ACSU that recursively computes PMs and output decision bits for each possible state transition. After that the decision bits are stored in and retrieved from the SMU in order to decode the source bits along the final survivor path. T-algorithm requires extra computation in the ACSU loop for calculating optimal PM. So It automatically reduce the decoding speed.

For the decoding of convolutional codes we can observe two types soft decision decoding and hard decision decoding. Soft decision decoding is much complex at which we are not concentrating at our research.

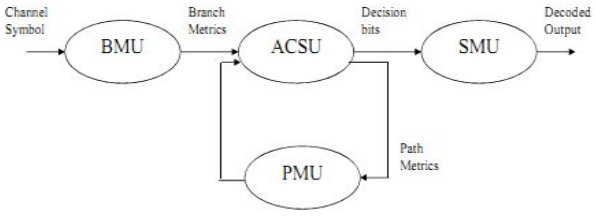


Figure 2: Viterbi decoder block diagram.

Computing the Branch Metric (BM)

The branch metric is a measure of the distance between what was transmitted and what was received and is defined each arc in the trellis. In hard decision decoding, where we have given a sequence of parity bits, the branch metric is the hamming distance between the expected parity bits and the received bits.

As example is shown in Fig. 4.11 where received bits are 00. For each state transition, the number on the arc shows branch metric for its transition. Two of the branch metrics are 0, corresponding to only states and transitions where the corresponding hamming distance is 0.

An attractive soft decision metric is the square of the difference between the received and expected. If the convolutional code produces the p parity bits, and the corresponding analog samples are $v = v_1, v_2, v_3, \ldots, v_p$, then we can construct the branch metric as (1)

$$BM[U,V] = \sum_{i=1}^{p} (Ui - Vi)2 \qquad (1)$$

Where $u = u_1, u_2, u_3, \ldots, u_p$ are expected parity bits.

Computing the Path Metric (PM)

Suppose the receiver has computed the path metric PM[*s, i*] for each state *s* (of which there are 2^{k1} , where k is the constraint length) at time step i.

The value of PM[*s, i*] is the total number of bit errors detected when comparing the received parity bits to the most likely transmitted message, considering all messages that could have been sent by the transmitter until time step i (starting from state '00' , which we will take by convention to be the starting state always).

Among all possible states at time step *i,* the most likely state is the one with the smallest path metric. I f there are more than one state, they are all equally possibilities. Now we determine the path metric at time step *i+1,* PM[*s,i+1]* ,First observe that if the transmitter is at state *s* at time step *i+1,* then it must have been in only one of the two possible states at time step *i.* These two predecessor states, labeled α and β, are always same for a given state. In fact, they depend only on the constraint length of the code and not on the parity functions. Fig. 3.3.1.1.2 shows the predecessor states for each state. For instance, for state 00, α=00 and β=01; for state 01, α=10,β=11.

Any message sequence that leaves the in state s at time *i+1* must have left the transmitter in state α or in a state β at time *i* . To arrive in state '01' at time *i+1,* one of the two properties must hold are: The transmitter was in state '10' at time '*i*' and the i^{th} message bit was 0.

Then the path metric of the new state, PM ['01', *i+1]* is equal to PM ['10', *i+2],* because the new sate is '01' and the corresponding path metric is larger by 2 containing the two errors.

The other possibility is that the transmitter was in state'11' at time *i* and i^{th} message bi was 0. If that is the case, then the transmitter sent 01 as the parity bits and there was one bit error, because we received 00. The path metric of the new state, PM ['01', *i+1*] and PM['11', *i+1*], Finally we can construct the path metric as (2).

PM[*s,i+1*]=min(PM[α,i]+BM[α→s],PM[β,i]+BM[β→s]),(2) Where α and β are two predecessor sates

Add-Compare-Select

A new value of the state metrics has to be computed at each time instant. In other words, the state metrics have to be updated every clock cycle. Because of this recursion, pipelining, a common approach to increase the throughput of the system, is not applicable. The Add-Compare-Select (ACS) unit hence is the module that consumes the most power and area. In order to obtain the required precision, a resolution of 7 bits for the state metrics is essential, while 5 bits are needed for the branch metrics. Since the state metrics are always positive numbers and since only positive branch metrics are added to them, the accumulated metrics would grow indefinitely without normalization. In this project we have chosen to implement modulo normalization, which requires keeping an additional bit (8 instead of 7). The operation of the ACS unit is shown in Figure 3.3.1.1.3 The new branch metrics are added to previous state metrics to form the candidates for the new state metrics. The comparison can be done by using the subtraction of the two candidate state metrics, and the MSB of the difference points to a larger one of two.

Hamming distance between the received code word and the allowed code word is calculated by checking the corresponding bit positions of the two code words. For example hamming distance between the code words 00 and 11 is 0 or the hamming distance between the code words 00 and 11 is 2. The hamming distance metric is cumulative so that the path with the largest total metric is final winner. Thus the hard decision Viterbi decoding makes use of maximum hamming distance in order to determine the output of the decoder.

Viterbi Decoder with T-Algorithm

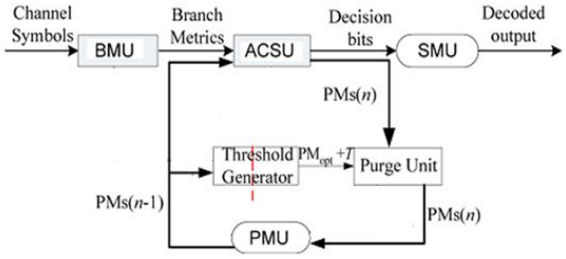


Figure 3: Viterbi decoder with T-algorithm.

The viterbi decoder with T–algorithm is shown in the figure 4.23 As compared with the Ideal viterbi decoder here we will have the extra computation like Threshold generator and purge unit. We will modify the path of ACSU by the extra computation of threshold generator and purge unit as shown in the figure 3.B.16. By using the Threshold generator the number of computations will decrease, by that the power consumption of entire system will decrease. The theoretical iteration bound also we will get.

Purge Unit:

When a state is purged in ACSU, the corresponding registers in SMU are not updated; thus, the power consumption is reduced. purged computation changes accordingly in the case of the T-algorithm on PMs, whereas in the case of the T–algorithm on BMs, the number of purged computation remains almost the same, regardless of the channel condition.

The rate ½ convolutional code employed in TCM system is shown in fig. 5 Preliminary bit error rate(BER) have been discussed . Estimated BER performance of the VD employing T-algorithm with different values of T over an additive Gaussian noise channel . The simulation is based on TCM employing the rate 1/2 code. Compared to the ideal viterbi algorithm, the threshold 'T_{pm}' can be lowered to 0.3 with less than 0.1 db performance loss.

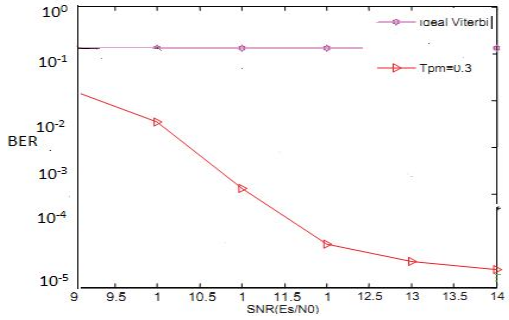


Figure 4: estimated BER performance with T-algorithm.

The functional block diagram of the VD with T-algorithm is shown in fig. The minimum value of each BM group (BMG) can be calculated in BMU or TMU and then passed to the 'Threshold Generator' unit (TGU) to calculate (PM_{opt}+T). (PM_{opt}+T) and the new PMs are then compared in the 'purge unit' (PU). The 64 states and PMs are labeled from 0 to 63. The precomputation steps is expressed as…

PM_{opt}=min[min{min(cluster0(n-2))+min(BMG0(n-1)),

min(cluster1(n-2))+min(BMG1(n-1)),

min(cluster2(n-2))+min(BMG3(n-1)),

min(cluster3(n-2))+min(BMG2(n-1))}+min(even BMs(n),

min{min(cluser0(n-2))+min(BMG1(n-1)),

min(cluster1(n-2)+min(BMG0(n-1)),

min(cluster2(n-2))+min(BMG2(n-1)),

min(cluster3(n-2))+min(BMG3(n-1))}+min(oddBMs(n))]

We can obtain (PM_{opt}+T) during the period when ACSU updates for new PMs. The only extra calculation for T-algorithm is the comparison between the (PM_{opt}+T) and all the PMs. Therefore the critical path is greatly shortened as the equation:

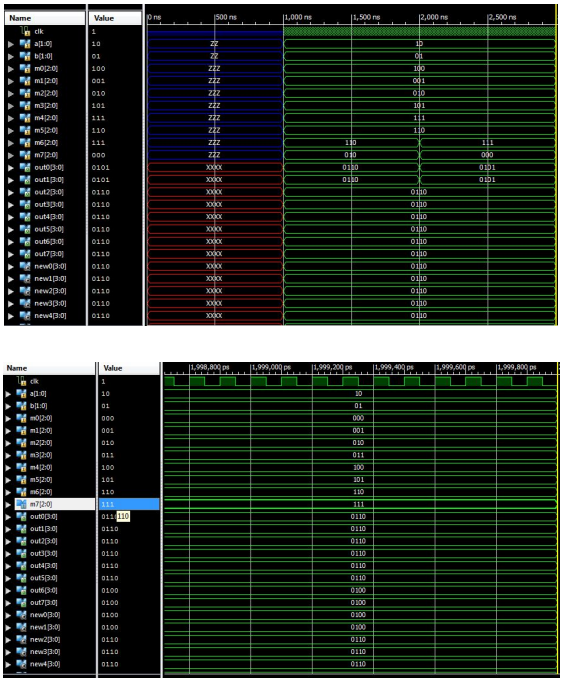
$T_{T\text{-}algo}=T_{adder}+T_{4\text{-}in\_comp}+2T_{2in\_comp}$

IV CONCLUSION

We have proposed a high-speed low-power VD design for TCM systems. The pre-computation architecture that incorporates T –algorithm efficiently reduces the power consumption of VDs without reducing the decoding speed appreciably. We have also analyzed the pre-computation algorithm, where the optimal pre-computation steps are calculated and discussed. This algorithm is suitable for TCM systems which always employ high-rate convolutional codes. Finally, we presented a design case. Both the ACSU and SMU are modified to correctly decode the signal.

The different modules are designed using Verilog and synthesized using Xilinx Integrated software environment (ISE) .The maximum operating power achieving for this project is 81%.

V. RESULTS



REFERENCES

[1] Vasily P. Pribylov, Alexander I. Plyasunov (2005). “A Convolutional Code Decoder Design Using Viterbi Algorithm with Register Exchange History Unit”. SIBCON. IEEE.

[2] John G. Proakis (2001). “Digital Communication”. McGraw Hill, Singapore. S. K. Hasnain, Azam Beg and S. M. Ghazanfar Monir (2004).

[3] “Performance Analysis of Viterbi Decoder Using a DSP Technique”. INMIC IEEE.

[4] Irwin M. Jacobs (1974). “Practical Applications of Coding”. IEEE. pp 305-310.

[5] YOU Yu-xin, WANG Jin-xiang, LAI Feng-chang and YE Yi-zheng (2002). “VLSI Design and Implementation of High Speed Viterbi Decoder”. IEEE .pp 64-66.

[6] Rodger E. Ziemer, Roger L. Peterson: Introduction to digital communication; Prentice-Hall International (UK), cop. 2001